SQL

Inhalt

- Grundlagen
- Befehlskategorien
- Basis-Befehle
- Mengenoperationen

Übersicht

Grundlagen

- Projektion versus Selektion
- Subqueries
- Kreuzprodukt

Projektion versus Selektion

Projektion

- Wahl bestimmter Spalten (Attribute)

```
SELECT Vorname, Nachname FROM Kunden;
```

Selektion

- Wahl bestimmter Zeilen (Tupel)

```
SELECT * FROM Kunden WHERE Ort = 'Berlin';
```

Subqueries

- SQL-Abfragen, die **innerhalb einer anderen Abfrage** geschachtelt werden
- liefern Zwischenergebnisse, die von der äußeren Abfrage verwendet werden
- helfen dabei:
  - Komplexe Bedingungen zu formulieren
  - dynamische Vergleichswerte zu ermitteln
  - Daten in mehreren Schritten logisch zu verknüpfen

Art	Verwendung	Beispiel
In SELECT	Ermitteln eines Werts in einer Spalte	SELECT (SELECT COUNT(*) FROM Kunden)
In WHERE	Bedingungen mit dynamischen Werten	WHERE Preis > (SELECT AVG(Preis) FROM Artikel)
In FROM (Inline-View)	Temporäre Ergebnistabelle	FROM (SELECT ...) AS Teilabfrage
In HAVING	Gruppenergebnisse vergleichen	HAVING SUM(Betrag) > (SELECT ...)
Korrelierte Subquery	Nutzt Werte der äußeren Abfrage	WHERE EXISTS (SELECT ... WHERE a.ID = b.ID)

Kreuzprodukt/kartesisches Produkt

- Basis für Joins (inner/outer)

Das **kartesische Produkt** zweier Tabellen A und B enthält **alle möglichen Kombinationen** der Zeilen aus A mit den Zeilen aus B.

Wenn A *m* Zeilen und B *n* Zeilen enthält, dann hat das Ergebnis **m × n Zeilen**.

Übersicht

Befehlskategorien

- Überblick
- DDL
- DML
- DQL
- DCL
- TCL

Befehlskategorien

Abk.	Name	Aufgabe	Typische Befehle
DDL	Data Definition Language	Struktur der Datenbank definieren/verwalten	CREATE, ALTER, DROP, TRUNCATE
DML	Data Manipulation Language	Daten in Tabellen verändern	INSERT, UPDATE, DELETE, MERGE
DQL	Data Query Language	Daten abfragen	SELECT
DCL	Data Control Language	Zugriffsrechte steuern	GRANT, REVOKE
TCL	Transaction Control Language	Transaktionen verwalten	COMMIT, ROLLBACK, SAVEPOINT

DDL – Data Definition Language

- Verwaltet die Struktur (Schema) der Datenbank

- `CREATE TABLE Kunden (...)` – neue Tabelle erstellen
- `ALTER TABLE Kunden ADD Spalte` – Struktur ändern
- `DROP TABLE` – Tabelle löschen

DML – Data Manipulation Language

- Bearbeitet den **Inhalt** von Tabellen (Daten) - Datenmanipulation

```
• INSERT INTO Kunden VALUES (...)
• UPDATE Kunden SET Ort = 'Berlin'
• DELETE FROM Kunden WHERE ...
```

DQL – Data Query Language

- Dient der **Abfrage** von Daten – ohne sie zu ändern

```
SELECT * FROM Kunden WHERE Ort = 'Berlin'
```

DCL – Data Control Language

- Steuert **Zugriffsrechte und Sicherheit**

```
GRANT SELECT ON Kunden TO BenutzerX
REVOKE INSERT ON Kunden FROM BenutzerY
```

TCL – Transaction Control Language

- Verwaltet **Transaktionen** (Mehrschritt-Operationen)

- `BEGIN TRANSACTION`
- `COMMIT` – dauerhaft speichern
- `ROLLBACK` – Änderungen rückgängig machen

Übersicht

Basis-Befehle

- CRUD
  - Create
  - Select
  - Update
  - Delete
- Ausdrücke und Bedingungen
- WHERE versus HAVING

CRUD - Create

```
INSERT INTO Kunden (KundenID, Vorname, Nachname, Ort, EMail)
VALUES (3, 'Clara', 'Fischer', 'München', 'clara@example.de');
```

Tabelleninhalt „Kunden“:

KundenID	Vorname	Nachname	Ort	EMail
1	Anna	Meier	Berlin	anna@example.de
2	Clara	Fischer	München	clara@example.de

CRUD - Select

```
SELECT * FROM Kunden
WHERE Ort = 'Berlin';
```

Ergebnis:

KundenID	Vorname	Nachname	Ort	EMail
1	Anna	Meier	Berlin	anna@example.de

CRUD - Update

```
UPDATE Kunden
SET EMail = ‚anna.meier@web.de‘
WHERE KundenID = 1;
```

Tabelleninhalt „Kunden“:

KundenID	Vorname	Nachname	Ort	EMail
1	Anna	Meier	Berlin	anna.meier@example.de
2	Clara	Fischer	München	clara@example.de

CRUD - Delete

```
DELETE FROM Kunden
WHERE Nachname = 'Fischer';
```

Tabelleninhalt „Kunden“:

KundenID	Vorname	Nachname	Ort	EMail
1	Anna	Meier	Berlin	anna.meier@example.de

Ausdrücke und Bedingungen

- **Ausdrücke** (engl. *Expressions*)
  - Kombinationen aus Konstanten, Spaltennamen, Operatoren, Funktionen und/oder Werten, die zu einem Ergebnis führen

Ausdruck	Bedeutung
`Preis * Menge`	Rechenoperation (Multiplikation zweier Felder)
`ALTER(TIMESTAMP '2000-01-01')`	Datumsfunktion
`ROUND(Gehalt, 2)`	Runden auf zwei Nachkommastellen

- **Bedingungen** (engl. *Conditions*)
  - bestimmen, **ob eine Zeile einer Tabelle** für eine bestimmte Operation berücksichtigt wird

Bedingung	Bedeutung
`Gehalt > 3000`	Zeilen mit Gehalt über 3000
`Name LIKE 'A%'`	Namen, die mit A beginnen
`Status IS NULL`	Zeilen ohne Wert im Feld `Status`
`Geburtsdatum BETWEEN '2000' AND '2010'`	Geburtsdatum zwischen 2000 und 2010
`Land IN ('DE', 'AT', 'CH')`	Land ist eines der drei Länder

WHERE versus HAVING

WHERE

- Prüft einzelne Zellenwerte

```
WHERE Gehalt > 2000
```

HAVING

- In Verbindung mit Gruppierungen/Aggregatfunktionen

```
HAVING min(Gehalt) > 2000
```

Übersicht

Mengenoperationen

- Schnittmenge
- Vereinigungsmenge
- Differenzmenge
- Sortierung
- Gruppierung
- Aggregatfunktionen
- JOIN

Schnittmenge - INTERSECT

- Gibt nur Datensätze zurück, die in **beiden** SELECT-Abfragen vorkommen

```
SELECT Name FROM Kunden
INTERSECT
SELECT Name FROM NewsletterEmpfänger;
```

Gibt alle Kunden aus, die auch im Newsletter-Verteiler sind.
Voraussetzung: Die SELECTs müssen die **gleiche Anzahl und Reihenfolge** von Spalten mit **kompatiblen Datentypen** haben.

Vereinigungsmenge - UNION (ALL)

- Gibt **alle eindeutigen** Datensätze zurück, die in einer der SELECT-Abfragen vorkommen

```
SELECT Name FROM Kunden
UNION
SELECT Name FROM Lieferanten;
```

Gibt eine Liste aller Namen aus beiden Tabellen – ohne Duplikate.

`UNION ALL` -> alle Datensätze inkl. Duplikate

Differenzmenge – MINUS (EXCEPT)

- Gibt alle Datensätze aus der **ersten SELECT-Anfrage**, die **nicht in der zweiten** enthalten sind

```
SELECT Name FROM Kunden
MINUS
SELECT Name FROM NewsletterEmpfänger;
```

Gibt alle Kunden aus, die **nicht** im Newsletter-Verteiler sind.

Sortierung

- Legt fest, in welcher Reihenfolge die Abfrageergebnisse angezeigt werden

- **Befehl:** `ORDER BY`
- Optionen: auf- oder absteigend (`ASC`, `DESC`)

Gruppierung

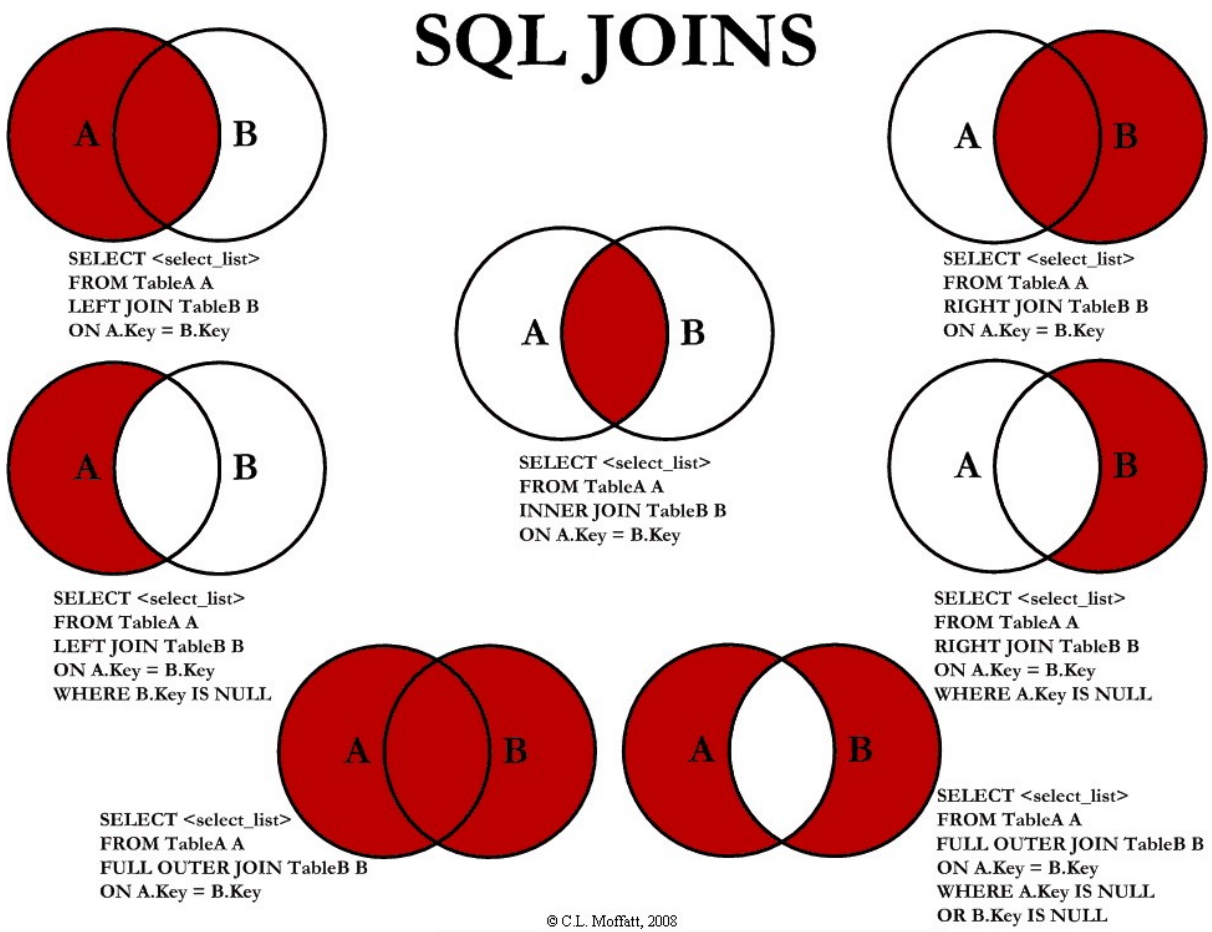
- Dient dazu, Datensätze anhand gemeinsamer Merkmale zusammenzufassen
- Häufig in Kombination mit Aggregatfunktionen

- **Befehl:** `GROUP BY`
- Wird oft mit `HAVING` verwendet, um Gruppenergebnisse zu filtern

Aggregatfunktionen

Funktion	Beschreibung	Beispielverwendung
COUNT()	Zählt die Anzahl der Zeilen (oder Nicht-NULL-Werte)	`COUNT(*)` alle Zeilen zählen
SUM()	Bildet die Summe aller Werte in einer Spalte	`SUM(Betrag)` Gesamtumsatz berechnen
AVG()	Errechnet den Durchschnittswert	`AVG(Preis)` durchschnittlicher Preis
MIN()	Gibt den kleinsten Wert in einer Spalte zurück	`MIN(Geburtsdatum)` älteste Person finden
MAX()	Gibt den größten Wert in einer Spalte zurück	`MAX(Gehalt)` höchstes Gehalt
GROUP_CONCAT() (in MySQL)	Verbindet mehrere Zeilen als kommagetrennten Text	`GROUP_CONCAT(Name)` Namen zusammenführen

JOIN


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008